## REMARKS

Reconsideration and further examination is respectfully requested.

The Applicant notes that the Examiner has rejected claims 1-3 and 14-15 based on Patent Application No. US 2002/0118682 (Choe et al.) under 35 U.S.C. § 102(e). Withdrawal of the rejection is respectfully requested.

Choe et al. lists a U.S. filing date of December 20, 2001. This date, however, is after the filing date of the present application, which was filed on November 9, 2001. Thus, the effective filing date of Choe et al. must be either the foreign application priority date of February 15, 2001, based on a Korean patent application, or the filing date of U.S. Provisional Application filed on December 22, 2000.

The foreign priority date does not qualify as an effective filing date for purposes of prior art. Under 35 U.S.C. §102(e), in order to rely on the foreign priority date the foreign application must be a WIPO publication under PCT article 21(2), be based on a PCT application designating the United States, and published in English. MPEP § 706.02(f)(1). The foreign application in question is not a PCT International application, and therefore cannot claim the benefit of any early filing date as a prior art reference.

Accordingly, the effective filing date of the reference is the provisional filing date of December 22, 2000. In this regard, the MPEP states "[i]f the application properly claims benefit under 35 U.S.C. 119(e) to a provisional application the effective filing date is the filing date of the provisional application for claims which are fully supported under the first paragraph of 35 U.S.C. 112 by the provisional application. MPEP § 706.02(V)(D).

Any rejection of claims of the present application based on Choe et al. must be based on disclosure contained in the provisional application to which the published patent application

claims priority. The provisional application is enclosed herewith, and a review of this reference shows that the rejection of claims 1-3 and 14-15 cannot be supported based on this disclosure. Portions of the published patent application relied on by the Examiner do not appear in the provisional application.

The Choe et al. provisional does not teach or suggest the use of an LC-trie algorithm for use in connection with an IP-address lookup table. In fact, the Choe et al. provisional actually teaches away from such an approach. Section 2 of the Choe et al. provisional discuss previous work, and begins with a review of existing router architecture. Then the reference discusses difficulties encountered in the prior work when it came to adding new routes and deleting old routes in IP-address lookup tables (page 5). Next, ensues a discussion of various data structures, including one based on the work of Nillson and Karlsoson (page 6, footnote 17). This is in fact the very work upon which the present application is based. Rather than adapting the LC-trie approach proposed in abstract by Nillson and Karlsoson, the Choe et al. provisional teaches against its use (page 6). In specific reference to the work of Nillson and Karlsoson, Choe et al. states "[a]lthough previously mentioned schemes contributed to give the reduction of lookup time, there still exists a problem concerning to [sic] the route updates." (page 6). Choe et al. then goes on to expressly abandon these prior approaches in favor of a different approach described as a randomized algorithm (page 6).

The Choe et al. provisional makes reference, and gives consideration, to the very same LC-trie compression model adapted and claimed in the present application, and rejects its use for the purpose of maintaining routing and forwarding tables of IP-addresses. Thus, the Choe et al. provisional patent application actually teaches <u>not</u> to use the LC-trie approach in combination with the forwarding and routing tables.

The approach favored by Choe et al. is actually quite opposite to that of the present invention. Choe et al. states "[t]he inverted key value sequence means that the search will be started from leaves to root unlike to [sic] the traverse sequence in a trie or a tree data structure" (page 7).

The Choe et al. provisional does not teach each element of the claimed invention, but merely recites the prior art and then teaches an invention that substitutes a completely different algorithm for IP-address lookup despite being aware of the underlying algorithm upon with the present invention is based. Accordingly, Applicant respectfully submits that the cited prior art does not establish a prima facie case of novelty under 35 U.S.C. § 102(e) as applied to claims 1-3 and 14-15, and therefore requests withdrawal of the refusal of these claims.

Applicants have made a diligent effort to place the claims in condition for allowance. However, should there remain unresolved issues that require adverse action, it is respectfully requested that the Examiner telephone Daniel A. Rosenberg, Applicants' Attorney at 515-288-2500 so that such issues may be resolved as expeditiously as possible.

For these reasons, and in view of the above amendments, this application is now considered to be in condition for allowance and such action is earnestly solicited.

Respectfully Submitted,

5-10-05
Date

Daniel A. Rosenberg
Attorney
Reg. No. 44308

Daniel A. Rosenberg
Attorney
Suite 2500, The Financial Center

666 Walnut Street
Des Moines, Iowa 50309-3993

Tel. 515-288-2500

Please type a plus sign (+) inside this box → [+]

PTO/SB/16 (8-10)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.





# PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request fro filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53 (c).

## INVENTOR(S)

| Given Name (first and middle [if any]) | Family Name or Surname | Residence (City and either State or Foreign Country) |
|---|---|---|
| MYONG-SU | CHOE | 5/5, 403 Bokwang-dong, Yongsan-gu, Seoul, 140-220, Republic of Korea |

☐ *Additional inventors are being named on the ___ separately numbered sheets attached hereto*

## TITLE OF THE INVENTION (280 characters max)

HIGH-SPEED IP ROUTING LOOKUPS AND ROUTING TABLE MANAGEMENT

## CORRESPONDENCE ADDRESS

Direct all correspondence to:

☐ Customer Number [ ] → Place Customer Number Bar Code Label here

*OR* *Type Customer Number here*

| | | | | | |
|---|---|---|---|---|---|
| ■ Firm or Individual Name | ROBERT E. BUSHNELL & LAW FIRM | | | | |
| Address | 1522 K Street, NW, Suite 300 | | | | |
| City | Washington | State | DC | ZIP | 20005 |
| Country | U.S.A. | Telephone | (202) 408-9040 | Fax | (202)628-0755 |

## ENCLOSED APPLICATION PARTS (*check all that apply*)

■ Specification *Number of Pages* 14
■ Drawing(s) *Number of Sheets*: 5
☐ Application Data Sheet. See 37 CFR 1.76
☐ CD(s), Number ____
☐ Other (specify): ____

## METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT (*check one*)

| | FILING FEE AMOUNT ($) |
|---|---|
| ☐ Applicant claims SMALL ENTITY status. See 37 CFR 1.27. | |
| ■ A check or money order is enclosed to cover the filing fees (Check #37663). | ■ $150.00 |
| ☐ The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 02-4943 | ☐ $75.00 |
| ☐ Payment by credit card. Form PTO-2038 is attached. | |

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.
■ No
☐ Yes, the name of the U.S. Government agency and the Government contract number are: ____

*Respectfully submitted,*
SIGNATURE R. E. Bushnell

Date: 12/22/00
REGISTRATION NO.: 27,774 *(If appropriate)*

TYPE or PRINTED NAME: Robert E. Bushnell, Esq.

TELEPHONE: (202) 408-9040

Docket Number: P56293P

## USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is used by the public to file (and by the PTO to process) a provisional application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the complete c\provisional application to the PTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this from and/or suggestions fro reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S.> Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, D.C., 20231.

PTO/SB/17 (08-00)
Approved for use through 9/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

12/22/00 DC77 U.S. PTO

# FEE TRANSMITTAL

Patent fees are subject to annual revision.

| Complete If Known | |
|---|---|
| Application Number | *to be assigned* |
| Filing Date | 22 December 2000 |
| First Named Inventor | MYONG-SU CHOE |
| Examiner Name | n/a |
| Group/Art Unit | n/a |
| Attorney Docket No. | P56293P |

TOTAL AMOUNT OF PAYMENT ($) 150.00

## METHOD OF PAYMENT (check one)

1. ☐ The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number: 02-4943
Deposit Account Number: _____

☐ Charge Any Additional Fee Required Under 37 C.F.R. §1.16 and 1.17.

☐ Applicant claims small entity status. See 37 CFR 1.27

2. ■ **Payment Enclosed:**
(CHECK #37863)
■ Check ☐ Credit Card ☐ Money Order ☐ Other

## FEE CALCULATION

### 1. BASIC FILING FEE

| Large Entity Fee Code | Large Entity Fee ($) | Small Entity Fee Code | Small Entity Fee ($) | Fee Description | Fee Paid |
|---|---|---|---|---|---|
| 101 | 710 | 201 | 355 | Utility filing fee | $ |
| 106 | 320 | 206 | 160 | Design filing fee | $ |
| 107 | 490 | 207 | 245 | Plant filing fee | $ |
| 108 | 710 | 208 | 355 | Reissue filing fee | $ |
| 114 | 150 | 214 | 75 | Provisional filing fee | $ |

SUBTOTAL (1) ($) .00

### 2. EXTRA CLAIM FEES

| | Extra Claims | | Fee from below | | Fee Paid |
|---|---|---|---|---|---|
| Total claims | -20** = | | x | = | |
| Independent Claims | - 3** = | | x | = | |
| Multiple Dependent | | | | = | |

** or number previously paid, if greater; For Reissues, see below

| Large Entity Fee Code | Large Entity Fee ($) | Small Entity Fee Code | Small Entity Fee ($) | Fee Description |
|---|---|---|---|---|
| 103 | 18 | 203 | 9 | Claims in excess of 20 |
| 102 | 80 | 202 | 40 | Independent claims in excess of 3 |
| 104 | 270 | 204 | 135 | Multiple dependent claim, if not paid |
| 109 | 80 | 209 | 40 | ** Reissue independent claims over original patent |
| 110 | 18 | 210 | 9 | ** Reissue claims in excess of 20 and over original patent |

SUBTOTAL (2) ($) 0.00

## FEE CALCULATION (continued)

### 3. ADDITIONAL FEES

| Large Entity Fee Code | Large Entity Fee ($) | Small Entity Fee Code | Small Entity Fee ($) | Fee Description | Fee Paid |
|---|---|---|---|---|---|
| 105 | 130 | 205 | 65 | Surcharge-late filing fee or oath | $ |
| 127 | 50 | 227 | 25 | Surcharge-late provisional filing fee or cover sheet | $ |
| 139 | 130 | 139 | 130 | Non-English specification | $ |
| 147 | 2,520 | 147 | 2,520 | For filing a request for reexamination | $ |
| 112 | 920* | 112 | 920* | Requesting publication of SIR prior to Examiner action | $ |
| 113 | 1,840 * | 113 | 1,840* | Requesting publication of SIR after Examiner action | $ |
| 115 | 110 | 215 | 55 | Extension for reply within first month | $ |
| 116 | 390 | 216 | 195 | Extension for reply within second month | $ |
| 117 | 890 | 217 | 445 | Extension for reply within third month | $ |
| 118 | 1,390 | 218 | 695 | Extension for reply within fourth month | $ |
| 128 | 1,890 | 228 | 945 | Extension for reply within fifth month | $ |
| 119 | 310 | 219 | 155 | Notice of Appeal | $ |
| 120 | 310 | 220 | 155 | Filing a brief in support of an appeal | $ |
| 121 | 270 | 221 | 135 | Request for oral hearing | $ |
| 138 | 1,510 | 138 | 1,510 | Petition to institute a public use proceeding | $ |
| 140 | 110 | 240 | 55 | Petition to revive - unavoidable | $ |
| 141 | 1,240 | 241 | 620 | Petition to revive - unintentional | $ |
| 142 | 1,240 | 242 | 620 | Utility issue fee (or reissue) | $ |
| 143 | 440 | 243 | 220 | Design issue fee | $ |
| 144 | 600 | 244 | 300 | Plant issue fee | $ |
| 122 | 130 | 122 | 130 | Petitions to the Commissioner | $ |
| 123 | 50 | 123 | 50 | Petitions related to provisional applications | $ |
| 126 | 240 | 126 | 240 | Submission of Information Disclosure Statement | $ |
| 581 | 40 | 581 | 40 | Recording each patent assignment per property (Times number of properties) | $ |
| 146 | 710 | 246 | 355 | Filing a submission after final rejection (37 C.F.R. §1.129(a)) | $ |
| 149 | 710 | 249 | 355 | For each additional invention to be examined (37 C.F.R. §1.129(b)) | $ |
| | | | | Other Fee (specify) 114, 1.16(k) Provisional application filing fee | $150.00 |
| | | | | Other Fee (specify) _____ | $ |

** Reduced by Basic Filing Fee Paid

SUBTOTAL (3) $150.00

## SUBMITTED BY

| Typed or Printed Name | Robert E. Bushnell, Esq. | Reg. Number | 27,774 |
|---|---|---|---|
| Signature | /Robert E. Bushnell/ | Date | 22 December 2000 |
| | | Deposit Account User ID | |

REB/kf

**WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.**

# High-speed IP routing lookups and routing table management

*Myongsu Choe*
Samsung Electronics Company LTD.
Sungnam-Shi, Kyunggi-Do, S. Korea
Email : mschoe@samsung.co.kr



**Abstract**

IP route lookup and its table management have been considered as one of the major bottlenecks in the design of a high-speed router. We present, in this paper, a randomized algorithm which proves to be efficient in the lookups and management of an IP routing table (or forwarding table) in comparisons with other trie-based algorithms. The algorithm is intended to use in a distributed or a parallel router architecture where a forwarding engine on each line card module or a forwarding engine separately located from a line card module is processing incoming packets apart from a routing processor in which the routing table is computed and maintained.

We compare complexity of our proposed algorithm to other ones. Our analytical results indicate that a variant of the randomized algorithm, a skip list with $k$-ary hashed nodes, significantly reduces the number of memory accesses required to update route changes and find a route with the longest prefix matching from the routing table (or forwarding table) as well as the amount of memory space to store route entries.

Key words: IP route lookups, longest prefix matching, forwarding engine, randomized algorithm.

## 1 Introduction

According to the increasing number of Internet users, variety of supported services, and the expansion of service areas such as VoIP and stream-oriented applications, the traffic in the Internet has been increased exponentially. Forwarding packets to the next hop interface by finding a destination path without causing any delay in a high-speed router has been emerged as a contingent design issue. In order to find the destination path,

managing a routing (or forwarding) table in a compact form and reducing lookup time are required.

In a traditional router before the advent of recently deployed high-speed routers where a required time to process packets and find their destinations is faster than the one on the transmission paths, a router as a relaying node connects subnetworks or other networks has played its roles well. Recently, the bandwidth increase of optical networking interface such as IP over DWDM surpassed processing time in a router and raised a blame that a router causes a dominant bottleneck in a high-speed Internet [1, 2, 3, 4, 5].

In addition, an introduction of a new IP addressing scheme called CIDR (Classless Inter-Domain Routing) [7, 8] to use IP addresses efficiently to prevent its address space from depleting proposed a different matching scheme called an LPM (Longest Prefix Matching) instead of an exact matching. Several algorithms usually using trie (or Patricia trie) has been widely used. This paper propose a longest prefix matching algorithm based on a randomized approach in which it will be used in the case of IPv6 deployment.

In this paper, we introduce a variant of the randomized algorithm which can provide minimal lookup time and an efficient maintenance of up-to-date routing table by reflecting routes changed from the routing processor due to the route flaps from neighboring routers. Furthermore, our scheme is planned to apply an 80-320 Gbps IP Switch which is developing by Samsung Electronics Company.

The remainder of this paper is organized along the following lines. Section 2 contains a description of our proposed high-speed router architecture as well as the previous route lookup work, section 3 describes our proposed algorithm, section 4 compares complexity of our algorithm and other typical ones, and the final section contains our conclusions.

## 2 Previous work

The recently announced or deployed backbone routers have been going on a distributed architecture as shown in Figure 1 to provide efficient packet processing in comparison with a centralized one where all the incoming packets from line cards are only processed in a routing processor.

A router consists of line card modules with a forwarding engine respectively, a routing processor and a switch fabric. For reliable operations, each different card can have its own redundant one as a standby purpose. Packets coming from other neighboring router(s) pass through the line card to switch fabric. The routing processor initially builds up its routing table and maintains it by reflecting changed routes. The switch fabric switches packets from input line cards to output ones and vice versa, *i.e.*, relaying packets between input and output line cards.

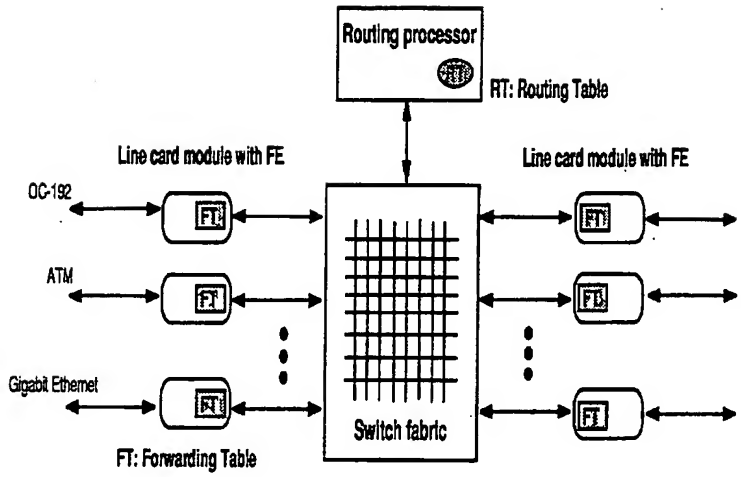


Figure 1: Distributed router architecture

Chan *et al.* [3] classifies two major lines of high-speed router's architecture as distributed or parallel one where its major distinction between them is dependent upon the location of the forwarding engine. As shown in Figure 1, the forwarding engine is embedded in a line card in which it connects other network node via a network interface. Almost commercial high-speed routers such as Cisco's Catalyst 8510 and PowerRail 5200 from Packet Engines, Inc., GigaRouter from NetStar including Samsung's Galaxy have the similar architecture as shown in Figure 1. On the other hand, Some experimental models such as Bell Labs [6] and BBN [4] followed on the track of the parallel architecture in Figure 2.

As the forwarding engine is separated from the line card, the parallel architecture uses a client and server based model for handling route lookups, parallel packet processing in each forwarding engine. In addition, there is a possibility to provide scalable packet processing capability as implied to mathematical analysis by Chan *et al.* [3]. Samsung also constructed a combined scheme in Figure 3 in which it integrates the parallel architecture with the distributed one, thereby providing cost-effective solutions by assigning a forwarding engine in a a line card with high-speed data link and by sharing a forwarding engine into a few line cards with low-speed data links.

The recently updated routing table generated from routing protocols such as RIP, OSPF or BGP-4 exists in the routing processor. A compact table called a forwarding one designed for an efficient lookup is copied from the routing table in a routing processor. The forwarding table is only designed for an efficient lookup by sacrificing the efficiency of route additions and deletions.

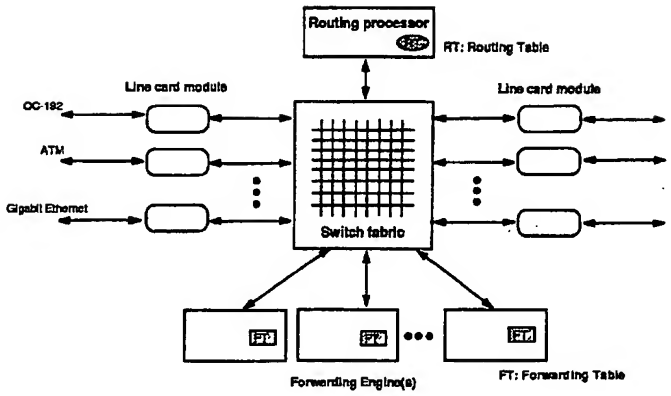


Figure 2: Parallel router architecture

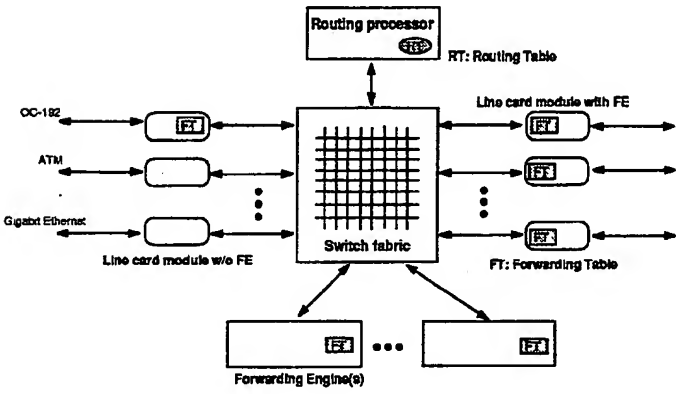


Figure 3: A combined scheme of Distributed and Parallel router architecture

If an incoming packet from an input link cannot find its destination path from its forwarding table, the corresponding packet must pass through the switch fabric to the routing processor to resolve its unmatched route. After finding its destination, then the packet must route to the switch again to transfer the packet to the output line card module. Otherwise, the packet is silently discarded at the routing processor due to the non-existing destination. The details of a forwarding engine and a routing processor showing the packet flow are also depicted in Figure 4 and 5.

The separation of routing table and forwarding one has been designed for the purpose of high speed packet processing where the routing table must be maintained to reflect the recent route changes. On the other hand, the forwarding table has been focused to support efficient lookup operation and to keep the whole forwarding table within its cache

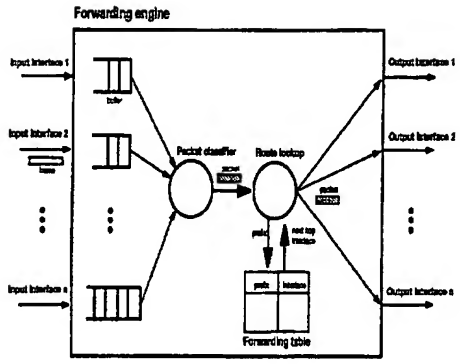


Figure 4: Structure of a forwarding engine

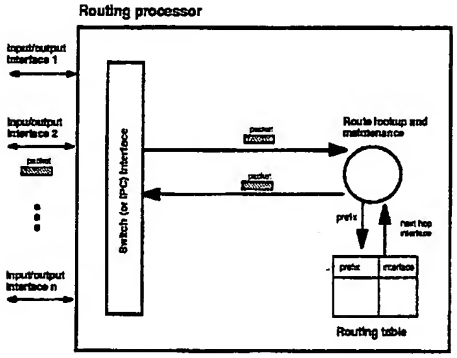


Figure 5: Structure of a routing processor

as compact as possible. Although the forwarding table (or routing one) is designed for an efficient LPM matching and the compactness of the table, adding new routes or deleting obsolete ones is inefficient.

Srinivasan and Varghese [11] showed interesting results by analyzing empirical data collected by IPMA [24]. Added or canceled routes in a day by a BGP routing protocol at a transit AS domain amounts to be 1,899,851 from the data recorded on Nov. 1, 1997. It means that a backbone router should perform 25 prefix updates/sec. We infer from the data that a data structure of a routing (or forwarding) table must be carefully designed to update route change promptly.

Typically used data structures in a routing table such as a radix tree or a Patricia trie caused increasing number of memory accesses to find a neighbor interface in order to transfer a packet to destination. Degermark *et al.* [10] proposed a compact forwarding

table capable to store in a cache of the forwarding engine except it is hard to reflect changed routes into it. A rope search algorithm [14, 15] by mapping a trie structure to a binary tree with hash tables and a complete prefix trie based on multi-resolution trie by Tzeng *et al.* [13] tend to be inefficient due to the update of changed route entries because those data structures are based on the trie. Aside from those ones, other variants of trie have been proposed. For example, a two-trie scheme [16] by linking two tries to reduce search time and an LC-trie [17] to reduce level length in a trie, and a DP-trie [9] were suggested.

Although previously mentioned schemes contributed to give the reduction of lookup time, there still exists a problem concerning to the route updates. Hardware-assisted schemes are also suggested to reduce lookup time. Gupta *et al.* [19] proposed a solution based on the use of large scale memory. Reducing lookup time is possible in comparisons with software-based ones, but it still poses significant amount of memory use and cost in the transition of IPv6. A scheme using CAM (Content Addressable Memory) was suggested by McAuley *et al.* [20], but due to the high price of CAM, it is not considering to use it at present. Huang *et al.* [21] proposes an indirect lookup algorithm using a pipelined memory access to reduce memory accesses although it has a disadvantage over IPv6 transition.

In this paper, we suggest an algorithm based on a randomized approach in which it will allow minimal memory accesses and easy maintenance of a routing or a forwarding table. In the case of route entry maintenance, it will reflect only changed route(s) instead of constructing the whole table. In addition, some algorithms [10, 11, 12, 15] demanded routes of prefix to be sorted prior to the table construction. On the other hand, our algorithm can be used to construct the corresponding table without any sorting in advance.

# 3 Randomized algorithm

A randomized algorithm is one that makes random choices during its execution. The behavior of such an algorithm may be random even on a fixed input. It focuses on establishing that it is likely to behave well on every input. The advantages of its use come from its simplicity and efficiency. We propose a variant of a skip list proposed by Pugh [22] for supporting efficient lookup and managing a routing table with ease in a high-speed router.

A skip list is considered as a linked list with sorted $n$ nodes and is usually used in the data structure on behalf of a balanced tree such as splay tree [23]. Thus it is simpler and more efficient than the balanced tree for insert and delete operations. We propose a variant of skip list for the longest prefix matching as shown in Figure 6.

Constructing a routing table or inserting changed routes into it can be performed with-

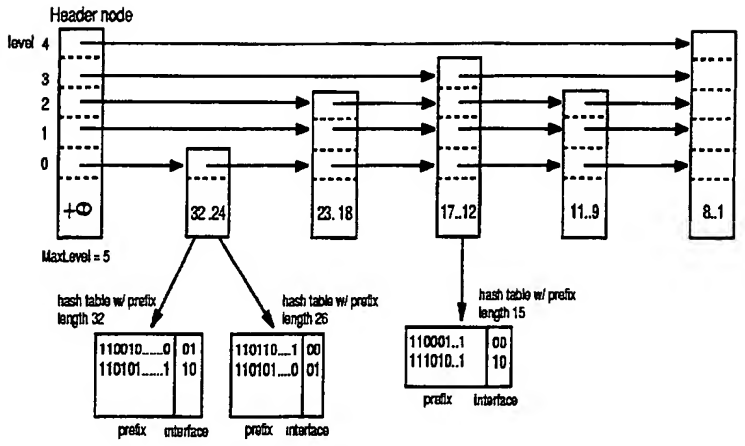


Figure 6: A variant of skip list with $k$-ary hashed nodes

out any distinctions. Each node has a single or multiple pointers and we name the leftmost node as a header node because all the operations on the variant of skip list such as lookup, insertions, deletions, and updates must be started from the node.

As shown in Figure 6, the header node contains $+\infty$ as a key value and a pointer to indicate next node. A key in each node means prefix length range or clustering of prefix length value and the key is inserted in a descending order. The inverted key value sequence means that the search will be started from leaves to root unlike to the traverse sequence in a trie or a tree data structure. The key value in the first node except the header one has the corresponding prefix range "32-24," and the route entry falling between the prefix length range is stored at one of the hash tables matching to its prefix length.

In Figure 6, the prefix length range is divided into a fixed way but variable length division scheme also expects to be possible. Srinivasan and Varghese [11] used a dynamic programming technique to divide the prefix range to provide optimal storage. In our proposed algorithm, we use fixed prefix range division. For the case of IPv4, if the prefix length range is fixed to 8, then the total number of inserted nodes can be $5 * \lceil (1 + 32/8) \rceil$. As shown from the empirical result from [24], the distribution of actual referenced prefix length has a skewed one, *i.e.*, implying the locality pattern existing in an actual referenced prefix length distribution. Therefore, we may expect to use smaller number of nodes.

A variable $MaxLevel$ represents the maximum level number among all the nodes belonging to the skip list, *i.e.*, the total number of pointers at the header node to point other nodes. Each node keeps a key value $key(x)$ and pointer(s) $next(x, L)$ to denote a pointer from the node $x$ to point a neighboring node at the level $L$ of node $x$.

We let each key in a node have prefix length range. In Figure 6, the first node except the header one has route entries ranging from 32 to 24 and each prefix length has its own hash table to store route entries matching to the corresponding prefix. If there exist all the route entries with prefix ranging from 32 to 24, then the corresponding node has 9 pointers to point 9 different hash tables. Each hash table only stores route entries matching its prefix length exactly. That is why we call a variant of skip list as a skip list with $k$-ary hashed node(s).

We represent insert, lookup, delete, and update operations respectively as pseudo code in Figure 7,8,9,10.

# 4 Complexity analysis

Actually, it is not easy to compare all of the algorithms on the same platform because some of the algorithms were experimented in various computing environments. Therefore, it is desirable to compare our algorithms with other ones in terms of complexity [25].

Assume that $N$ route entries exist in a routing table. To represent an IP address, $W$ address bits are required and $k$ is defined as an algorithm dependent constant. Table 1 shows compared complexity of our algorithm and other ones in terms of time and memory space.

Table 1: Comparisons of Complexity

| | *Build* | *Insert* | *Delete* | *LPM Search* | *Memory* |
|---|---|---|---|---|---|
| Array | $O(N)$ | $O(N^2)$ | $O(N^2)$ | $O(NW)$ | $O(N)$ |
| Binary search tree | $O(NlgN)$ | $O(N)$ | $O(N)$ | $O(lg(2N))^a$ | $O(N)$ |
| Trie | $O(NW)$ | – | – | $O(W)$ | $O(NW)$ |
| Radix trie | $O(NW)$ | – | – | $O(W)$ | $O(N)$ |
| PATRICIA trie | $O(NW)$ | – | – | $O(W^2)$ | $O(N)$ |
| DP trie | $O(NW)$ | $O(W)$ | $O(W)$ | $O(W)$ | $O(N)$ |
| LC trie | $O(NW)$ | – | – | $O(W)$ | $O(N)$ |
| Hashed radix tree | $O(NW)$ | – | – | $O(W/k)$ | $O(NW)$ |
| Basic BST scheme without backtracking | $O(NlgW)$ | $O(lgN)$ | $O(lgN)$ | $O(lgW)$ | $O(NlgW)$ |
| Asymetric BST | $O(NlgW)$ | $O(N)$ | $O(N)$ | $O(lgW)$ | $O(NlgW)$ |
| Rope search | $O(NW)$ | $O(N)$ | $O(N)$ | $O(lgW)$ | $O(NlgW)$ |
| Ternary CAMs | $O(N)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(N)$ |
| Complete prefix tree | $O(NlgW)$ | – | – | $O(W/k)$ | $O(N)$ |
| Binary hash table search | $O(NlgW)$ | – | – | $O(lg(W))$ | $O(N)$ |
| Multiway and multicolumn search | $O(NW)$ | $O(N)$ | $O(N)$ | $O(lg_kN)$ | $O(N)$ |
| Large memory architecture | $O(N)$ | – | – | $O(W/k)$ | $O(N)$ |
| Skip list | $O(NlgN)$ | $O(lgN)$ | $O(lgN)$ | $O(NlgN)$ | $O(N)$ |
| Ours | $O(Nlg(W/k))$ | $O(lg(W/k))$ | $O(lg(W/k))$ | $O(lg(W/k))$ | $O(N)$ |

[a]Tzeng *et al.* [13] analyzed its complexity as $O(lg(N)+W)$. On the other hand, Waldovogel *et al.* [14] showed it as $O(lg(2N))$.

To avoid confusion from other logarithms with a base $e$ or 10, $lg$ denotes a logarithm

```
var k: interger init 0; (* prefix length *)
  prefix: bitstring init b'0';
  interface: integer init 0;

  Insert(k, prefix, interface) (* inserting a route entry *)
     begin
         newLevel := RandomLevel();
         allocate Nodes[newLevel+1];
         while L ≥ 0
             if next(x, L) = null or key(next(x, L)) > k then
                 if newLevel ≥ L then Nodes[L] := x;
                 L := L-1;
             else
                 x := next(x,L);
             end if
         end while
         if key(x) ≠ k then (* not found *)
             n := MakeNode(k,newLevel);
             for i := 0 to newLevel
                 next(n,i) : = next(Nodes[i],i);
                 next(Nodes[i],i) : = n;
             end for
         end if
         hash-insert (prefix,interface)
     end
  RandomLevel() (* deciding next level randomly *)
     begin
         L := 0;
         r := Random (); (* r ∈ [0, 1] *)
         while r < p and L < MaxLevel
             L : = L+1;
         end while
         return L;
     end
```

Figure 7: Insert operation

```
Search(k, prefix) (* looking for a route with the longest prefix *)
  begin
    x := Header;
    L := MaxLevel;
    while key(x) > k and L ≤ 0
      if next(x, L) = null then
        L := L-1;
      else
        if key(next(x, L)) < k then
          L := L-1;
        else
          x := next(x,L);
        end if
      end if
    end while
    if L < 0 or key(x) ≠ k then
      return null;
    else
      return hash-get(prefix);
    end if
  end
```

Figure 8: Lookup operation

```
Delete(k, prefix, interface) (* deleting a route entry *)
  begin
    if search (k, prefix) then
      hash-delete (prefix, interface);
    else
      report error;
    end if
  end
```

Figure 9: Delete operation

```
Update(k, prefix, interface) (* updating a route entry *)
  begin
    if search (k, prefix) then
      hash-update (prefix, interface);
    else
      report error;
    end if
  end
```

Figure 10: Update operation

with base 2. In the case of IPv4, $W$ corresponds 32 and 128 for IPv6. The MAE-EAST known as routers in a NAP (Network Access Point) has around 50,000 entries. In this case, $k$ in a skip list with $k$-ary hashed nodes is 4 if prefix range is equal to 8. It amounts to be 16 for IPv6 case.

Our algorithm is better than the traditional search tree and tries (Patricia trie or LC-trie) in comparisons with routing table construction, route lookup, insert, and delete operations. The complexity results indicates that our proposed algorithm outperforms a pure skip list and even binary hash table search [14] and a complete prefix trie [13] known to be licensed in commercial routers.

# 5 Conclusion

In this paper we introduced a skip list with $k$-ary hashed nodes which can be applicable to the route lookup algorithm in a high-speed router. We show that our algorithm outperforms previously proposed ones which only emphasize the efficiency of route entry look-up and the compactness of its table rather than table build-up, route lookup and update.

- We propose an integrated scheme of parallel and distributed router architecture. It seems to be a cost effective solution by dedicating a forwarding engine to an interface with high-speed data link layer and sharing autonomous forwarding engine(s) with interfaces with low-speed data links. By using load balancing, it may improve router system throughput. We are exploiting this possibility of the architecture by mathematical analysis and simulation.

- We suggest to use routing and forwarding tables with the same data structure for easy maintenance.

- Our proposed algorithm can be used in IP routing areas.
- Our algorithm also can be used in IPv6.
- Among the main reasons to use a randomized algorithm in the areas of route lookup, table construction and its update, the algorithm is known to be operating well on every input from probabilistic analysis. Our complexity comparisons with other ones indicate that it is better than the other algorithms in the worst case comparisons.
- We used an evenly fixed division scheme of prefix length range to make reduction of nodes in a skip list. It will be worth efforts to exploit variable prefix length range (or clustering) division by adapting prefix length distribution with locality pattern as implied from IPMA's actual gleaned data.
- By descending key value in each node of a skip list, the lookup operation starting from a header node searches the longest prefix matching from the leaves opposite to the case of a trie or a tree. Thus backtracking will not happen and a specially designed pointer called a marker [14] is not required, thereby expecting to save storage.
- Due to the characteristics of a skip list, there is no required for sorting in order to construct or insert route entries in a routing (or forwarding) table. On the other hand, some algorithms [10, 11, 12, 13, 14, 18] needs to be sorted by prefix length sequence to build up the table.
- As a fringe benefit, on the current trend of a contemporary architecture such as a distributed router architecture or a parallel one where a forwarding table in a forwarding engine still separate from a routing one in a routing processor, by transferring only updated routes along the paths into the router instead of the whole route entries, memory bandwidth or system bus contention may be relieved.
- In addition, the reduction of lookup time and the maintenance of the consistent routing information over all the routing related tables may contribute to the performance enhancement over other factors such as QoS, multicasting, IPSec, and others.

# References


[1] Keshav, S. and Sharma, R., "Issues and Trends in Router Design", *IEEE Communications Magazine*, pages 144-151,May,1998.

[2] Kumar, V. and Lakshman, T. and Stiliadis, D., "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet", *IEEE Communications Magazine*, pages 152-164,May,1998.

[3] Chan, H., Alnuweiri, H. and Leung, V., "A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers", *IEEE Journal of Selected Areas in Communications*, Vol. 17, No.6, pages 1013-1029, June 1999.

[4] Partridge, C. et al., "A 50-Gb/s IP Router", *IEEE/ACM Trans. on Networking*, vol. 6, no.3, pages 237-248, 1998.

[5] Metz, C., "IP Routers: New Tool for Gigabit Networking", *IEEE Internet Computing*, pages 14-18, Nov.-Dec.,1998.

[6] Asthana, A., Delph, C., Jagadish, H., and Krzyzanowski, P., "Towards a gigabit IP router", *J. High Speed Networks*, vol. 1, no.4, pages 281-288, ,1992.

[7] RFC 1518, "An Architecture for IP Address Allocation with CIDR", IETF, Sept., 1993.

[8] RFC 1517, "Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)", IETF, Sept., 1993.

[9] Doeringer W., Karjoth, G. and Nassehi, M., "Routing on Longest-Matching Prefixes", *IEEE/ACM Trans. on Networking*, vol.4, no.1, pages 86-97, Feb., 1996.

[10] Degermark, M., Brodnik, A., Carlsson, S. and Pink, S., "Small Forwarding Tables for Fast Routing Lookups", In *Proceedings of ACM SIGCOMM '97*, pages 3-14, Cannes, France, 1997.

[11] Srinivasan, V. and Varghese, G., "Faster IP Lookups using Controlled Prefix Expansion", In *Proceedings of ACM Sigmetrics '98 Conf.*, pages 1-11, 1998.

[12] Lampson, B., Srinivasan, V. and Varghese, G., "IP Lookups using Multiway and Multicolumn Search", In *Proceedings of IEEE Infocom Conf.*, pages 1248-1256, 1998.

[13] Tzeng, H. and Pryzygienda, T., "On Fast Address-Lookup Algorithms", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, pages 1067-1082, June, 1999.

[14] Waldvogel, M., Varghese, G., Turner, J. and Plattner, B., "Scalable High Speed IP Routing Lookups", In *Proceedings of ACM SIGCOMM '97*, Cannes, France, pages 25-37, 1997.

[15] Waldvogel, M., Varghese, G., Turner, J. and Plattner, B., "Scalable Best Matching Prefix Lookups", In *Proceedings of PODC '98*, Puerto Vallarta, page , 1998.

[16] Kijkanjanarat, T. and Chao, H., "Fast IP Lookups Using a Two-trie Data Structure", In *Proceedings of Globecom'99*, pages , 1999.

[17] Nillson, S. and Karlsoson, G., "IP-Addresses Lookup Using LC-Tries", *IEEE Journal on Selected Areas in Communications*, Vol.17, No. 16, pages 1083-1092, 1999.

[18] Crescenzi, P., Dardini, L. and Grossi, R., "IP Address Lookup Made Fast and Simple", Technical Report TR-99-01, Dipartimento di Informatica, University à Di Pisa, 1999.

[19] Gupta, P., Lin, S. and McKeown, N., "Routing Lookups in Hardware at Memory Access Speeds", In *Proceedings of IEEE INFOCOM '98 Conf.*, pages 1240-1247, 1998.

[20] McAuley, A. and Francis, P., "Fast Routing Table Lookup Using CAMs", In *Proceedings of IEEE INFOCOM '93*, Vol.3, pages 1382-1391, 1993.

[21] Huang, N. and Zhao, S., "A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, pages 1093-1104, June, 1999.

[22] Pugh, W., "Skip Lists: A Probabilistic Alternatives to Balanced Trees", *CACM 33(6)*, pages 668-676, 1990.

[23] Sleator, D. and Tarjan, R., "Self-Adjusting Binary Search Trees", *JACM*, vol. 32, no. 3, 1985.

[24] IPMA (Internet Performance Measurement and Analysis), http://nic.merit.edu/ipma.

[25] Cormen, T., Leiserson, C. and Rivest, R., *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

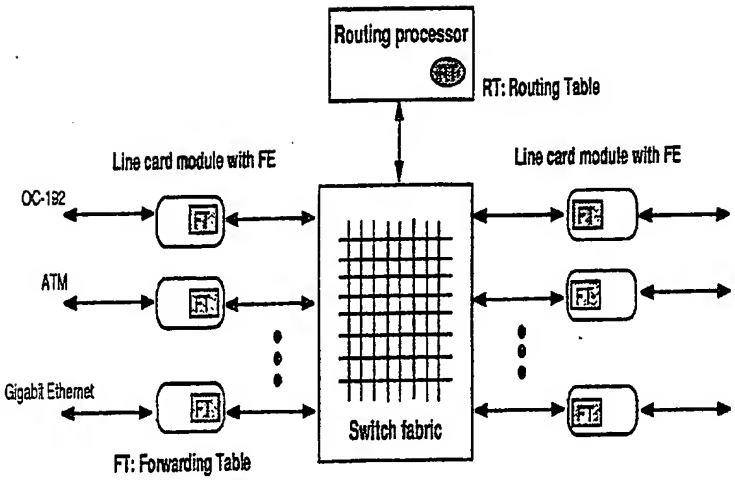


Figure 1: Distributed router architecture

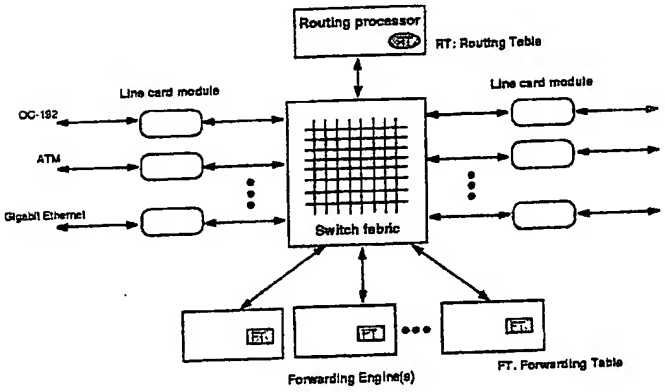


Figure 2: Parallel router architecture

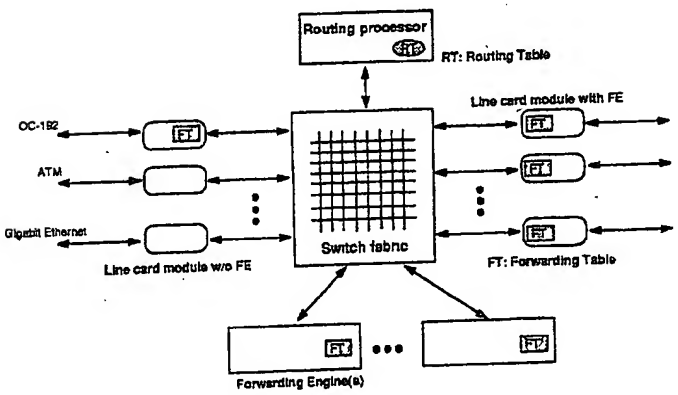


Figure 3: A combined scheme of Distributed and Parallel router architecture

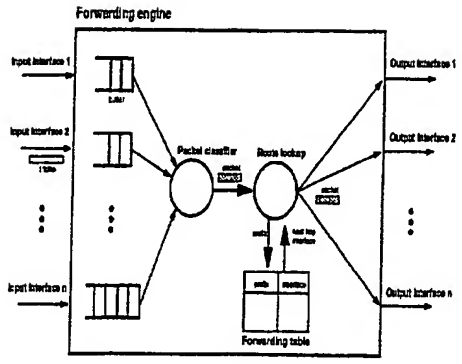


Figure 4: Structure of a forwarding engine

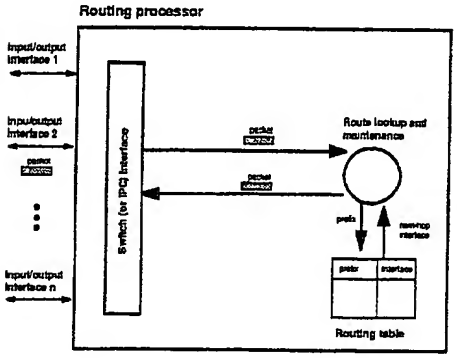


Figure 5: Structure of a routing processor

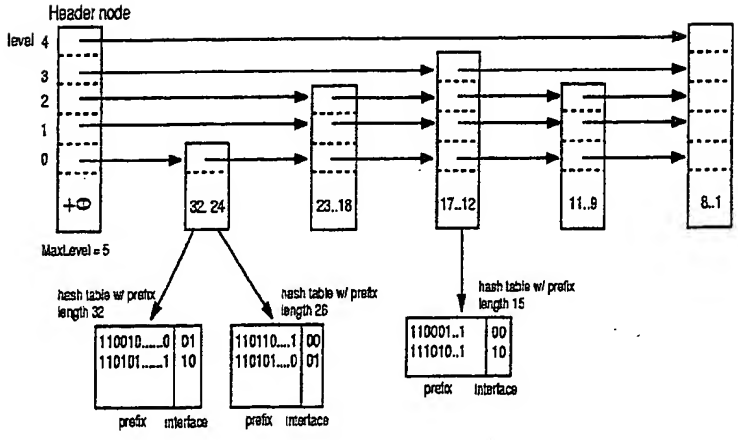


Figure 6: A variant of skip list with $k$-ary hashed nodes

```
var k: interger init 0; (* prefix length *)
  prefix: bitstring init b'0';
  interface: integer init 0;

  Insert(k,prefix,interface) (* inserting a route entry *)
     begin
        newLevel := RandomLevel();
        allocate Nodes[newLevel+1];
        while L ≥ 0
           if next(x,L) = null or key(next(x,L)) > k then
              if newLevel ≥ L then Nodes[L] := x;
              L := L-1;
           else
              x := next(x,L);
           end if
        end while
        if key(x) ≠ k then (* not found *)
           n := MakeNode(k,newLevel);
           for i := 0 to newLevel
              next(n,i) : = next(Nodes[i],i);
              next(Nodes[i],i) : = n;
           end for
        end if
        hash-insert (prefix,interface)
     end
  RandomLevel() (* deciding next level randomly *)
     begin
        L := 0;
        r := Random (); (* r ∈ [0,1] *)
        while r < p and L < MaxLevel
           L : = L+1;
        end while
        return L;
     end
```

Figure 7: Insert operation

```
Search(k, prefix) (* looking for a route with the longest prefix *)
  begin
    x := Header;
    L := MaxLevel;
    while key(x) > k and L ≤ 0
      if next(x, L) = null then
        L := L-1;
      else
        if key(next(x, L)) < k then
          L := L-1;
        else
          x := next(x,L);
        end if
      end if
    end while
    if L < 0 or key(x) ≠ k then
      return null;
    else
      return hash-get(prefix);
    end if
  end
```

Figure 8: Lookup operation

```
Delete(k, prefix, interface) (* deleting a route entry *)
  begin
    if search (k, prefix) then
      hash-delete (prefix, interface);
    else
      report error;
    end if
  end
```

Figure 9: Delete operation

```
Update(k, prefix, interface) (* updating a route entry *)
  begin
    if search (k, prefix) then
      hash-update (prefix, interface);
    else
      report error;
    end if
  end
```

Figure 10: Update operation